Ian Sinclair

ATOMIC RESEARCH

An in-depth probe into the increasingly popular ATOM. Does its language impediment really matter? Read on and find out.

I first saw the ACORN ATOM in operation early in 1980, and I was impressed with the speed at which the programs ran, and its excellent graphics commands. I was also impressed with the way in which the basic unit could be expanded onboard without having to buy expensive add-ons, because, to my mind, this is the feature which distinguishes the genuine baby computer from the throw-away toy.

When my ATOM kit arrived in September, I tore into the construction with considerable glee, and a record of my progress through both hardware and software may be of some interest. To keep the record straight, I have always maintained that computer kits are never particularly good value for money compared to the same computer in ready-to-go form, and that the small extra price asked for a fully assembled and tested item is always worthwhile. I still hold to that view, but I must admit that the ATOM kit was very well thought out, with a clear constructional guide. Everything, in fact, had been done to make construction easy, even for a relative beginner (after 35 years, I don't quite qualify there). The PCB in particular is an excellent piece of work, silkscreened with component outlines and part numbers, and with all the tracks, apart from mounting pads, coated with insulating varnish to prevent short circuits caused by splashes of solder.

I started assembly one Saturday morning with a new bit on the excellent little iron I use for all my constructional work, and the bit was all but gone by the time the last joint was made! Computer construction involves a lot of soldered joints, and wears out a lot of bits if you use ordinary 60/40 alloy!

Getting back to the kit, all the ICs, with the exception of the voltage regulators, are in sockets, so that soldering can be done using an unearthed iron. Some of the holders were on the tight side, so that plugging in the ICs later was not always easy — I would recommend any constructor to ease up all the socket holes of the holders with a needle or an old IC before putting in the delicate devices such as the 6502, the 8225, and the 2114 memory chips. Only about half of the board is actually populated with chips when the basic model is being built, but all the holders should be put in as this makes expansion so much easier — just a matter of plugging in more memory chips.

The most difficult part of assembly is the attachment of the keyboard. The keyboard connections are made through rather fine but springy wires which protrude a short distance beyond the underside of the keyboard. Each one of these has to be persuaded to pass through a generously-dimensioned hole in the PCB and, when all are in place, soldered to the rim of each hole. This is by no means easy, and I would very much have preferred a plug-and-socket arrangement, particularly in view of the fact that my keyboard developed a sticking key.

Up And Running

Sunday morning was switch-on day, and the portable telly was hooked up, turned on, and the ATOM attached. The ATOM uses an external transformer/rectifier unit which plugs into the board via a socket of the type used on some pocket calculators. The contact seemed rather dodgy to me, though it hasn't given any trouble except when the unit is shifted while working; on the whole I would have preferred a DIN type plug and socket for this task. Alas, though the screen filled with characters when switched on, indicating that life was present, the characters didn't clear when the BREAK key was pressed, revealing something fishy in the chips. Good as their word, though, Acorn sorted this out, along with the sticky key, at no charge.

Before we finally leave the construction stage, there's one feature I moan about on virtually every computer kit I come across. The stabilised 5V supply is obtained from a 7805 on board, and the manual suggests that a heatsink (a small piece of aluminium is supplied) is needed only if the ATOM is expanded (to a molecule?). Now, I have endless trouble with 7805's, more than with any other chip, and the cooler they run the happier I am. Even with the aluminium attached, and no expansion, the 7805 in the ATOM runs hotter than I like, and I would feel inclined to use either an external power supply (which is provided for in the ATOM design), or to cut the case so that I could use a fairly substantial finned heatsink. As it happens, I intend to expand my ATOM, so I shall run it from the excellent (though well-used) power supply I bought from Display Electronics some time ago.

Circulating The Electrons

With all the hardware sorted out, and the cassette loading and dumping checked using the very useful diagnostic method shown in the manual, it was time to start investigating the programming of the ATOM. People who have read the manual will tell you that it is a very strange versions of BASIC. It is indeed odd, to such an extent that I, having written both a series and a textbook on BASIC, still have to refer to the ATOM manual when I program. As far as the beginner is concerned, however, *all* computers are equally odd. The version of BASIC which is used by the ATOM is no more difficult (apart from string handling — see later) to learn than the Microsoft BASIC we all know and love — it's just different. There really isn't much point in simply saying that it's different without giving examples and looking at the reasons for the differences, so that the rest of this review will be devoted to a more detailed discussion of ATOM BASIC.

To start with, like the BASIC on several other very small computers, the ATOM BASIC is an Integer BASIC, which means that it doesn't handle fractions, decimal or otherwise. For many purposes this is unimportant to the beginner and the subject of using integers is well treated in the manual. It would be an obvious disadvantage for anyone who wanted to write programs for accounts, mathematical work or scientific analysis. As it happens, the ATOM would not be the best choice for anyone who intends writing such programs for a variety of other reasons, but an additional ROM chip can be added which provides a full range of mathematical functions, (see Tables 1&2) floating point arithmetic, (decimal fractions welcome) and colour graphics commands. Unfortunately, in order to make the expanded ATOM compatible with the unexpanded one, adding the extra ROM requires that some existing commands have to be altered to make use of the extra facilities. The alteration is the addition of F (for "floating") before some commands when the new ROM is to be used. For example, 'DIM' dimensions memory space for arrays or strings, but 'FDIM' has to be used for floating-point arrays. This example is fairly straightforward, but other commands such as 'FIF' and 'FINPUT' (floating-point IF and INPUT) are less so. Users who are not interested in the mathematical package need not worry about all this, and have the additional consolation that the ATOM will handle numbers between ±2 000 000 000 with complete accuracy, something that computers with floating-point numbers will not often do. The main point to remember with Integer BASIC is that divisions will give odd results because of the omission of the fractions.

The first instruction which figures prominently in any beginner's use of BASIC is PRINT and, on the face of it, the ATOM seems to use this in pretty much the same way as anything else, with a screen containing 16 lines of 32 characters each to print on. This apparent similarity is deceptive, however, because whereas on Microsoft BASIC, we need to use a command to get two PRINT statements on one line, the ATOM needs a command *not* to do this! For example, on my TRS-80, the commands:

```
10 PRINT "THIS IS"
20 PRINT "TRS-80"
```

will result in the output on the screen:

```
THIS IS
TRS-80
```

unless line 10 is written as: 10 PRINT "THIS IS"; ,using the semi-colon to indicate to the computer that PRINTing is to be continued on the same line. On the ATOM, lines such as:

```
10 PRINT "THIS IS"
20 PRINT " THE ATOM"
```

will print out as THIS IS THE ATOM unless a newline command (') is used, such as 10 PRINT "THIS IS" '. Note, incidentally, the space between the first quote mark and the T in line 20. If this space is omitted, then the T and the S of IS will be next to each other when PRINTed on one line.

To anyone brought up on Microsoft, this looks plainly perverse, but there is a very good reason for it which becomes apparent when you want to display tables on the screen. PRINTing, for example, 32 sets of numbers as four rows of eight columns is not completely straightforward with conventional BASIC, but on the ATOM it can be programmed by a line such as:

```
100 @ = 4;FOR N = 1 TO 32;PRINT
    AA(N);NEXT
```

The explanation for this is that @ = 4 sets up "fields" each of four character spaces, on the screen so that each number is fitted into a column. This allows a four character gap between the end of one column and the start of the next. The standard size, if we don't specify a value for @, is eight which gives us four columns. Using @ = 4 produces eight columns (with 32 characters per line there can only be eight lots of four), and the PRINT arrangements of the ATOM will therefore put the numbers into eight columns, printed in order to give four rows. No TAB command exists though there is a COUNT statement which keeps check of the numbers of characters which have been printed in a line. COUNT has to be used if the number of columns that you want to use will not divide into 32.

I have gone over the PRINT instruction in detail because it is the sort of difference between ATOM BASIC and Microsoft BASIC which could trip up even experienced programmers, although it is probably an advantage for the beginner who wants neat tabulation. No-one should have any difficulties over commands which are unique to the ATOM, because they are just new commands which can be learned, as we all have to learn new BASIC commands from time to time. The commands that are likely to give problems, mainly to people like me who may be programming different types of machine in the course of one day, are the ones which look like Microsoft BASIC but aren't, and those which look like nothing else on earth!

The Old And The New

Let's look at some examples. There's a command, OLD, which, when executed restores a program which had been NEW'd out. This is a command I would have given two ears and a tail for in the small hours of the morning when using some other machines. Provided that there was a program in the machine, and you haven't LOADed in another one (from cassette or keyboard), OLD will restore your program. Incidentally, if a program is stopped by using the ESC key, it will LIST normally, but if it is stopped by the BREAK key, it will not list unless the OLD routine is used first.

Other examples of "new" commands which are a useful enhancement of BASIC are the DO . . . . . UNTIL loop, and the PLOT, MOVE and DRAW instructions. The DO . . . . . UNTIL loop is a command which hasn't appeared on any other BASIC that I know of, but which is an essential part of Pascal and other structured languages. Its advantage is that it provides a neat solution to the problem of an indefinite loop, which has no satisfactory counterpart in ordinary BASIC. Suppose, for example, that we want to set up a number of subscripted variables, but we are uncertain of the

The rear panel connections on a standard ATOM.

number we need when the program is written. In conventional BASIC, we might write program sections such as:

```
10 N=1
20 INPUT A(N): IF A(N) = 0 THEN 40
30 N = N+1:GOTO 20
40 [next step after completing entry of
   array]
```

which needs two GOTO steps. As an alternative, we could use:

```
10 FOR N = 1 TO 100 : INPUT A(N)
20 IF A(N)=0 THEN N=N-1 ELSE NEXT
   N
30 [next step]
```

which looks neater, but commits the sin of jumping out of a FOR.....NEXT loop before the full allocation of steps (set at a maximum of 100 in line 10) has been performed. Not all versions of BASIC will allow this (because of the return address on the stack), though the TRS-80 is quite happy to permit such messy programming.

Using the DO.....UNTIL instruction, we can write lines such as:

```
5 DIM AA(100)
10 N=0; DO N=N+1;INPUT A;
   AA(N)=A
20 UNTIL A=0;N=N-1
```

which permits much neater programming of loops which have to be terminated at some count number or by some condition such as zero entry. Note, incidentally, that in ATOM BASIC, INPUT cannot be used to enter items into an array — a dummy variable has to be used, A in this example. In addition, each array has to be dimensioned.

Graphic Illustrations

The PLOT, DRAW and MOVE instructions are a gift for the keen graphics user and games nut. The PLOT instruction is a complicated one which has to be followed by three numbers, separated by commas. The first number is the one which demonstrates what the PLOT instruction will do — move the cursor position, draw a line, draw black on white or white on black, plot a point, etc. Using the PLOT command effectively needs a lot of experience, and a copy of the PLOT commands pinned up on the wall, but it permits a remarkable number of interesting graphic commands. The MOVE and DRAW commands are, in fact, simpler versions of PLOT which carry out one PLOT function each. MOVE means 'move a cursor (not visible) to a position on the screen', and DRAW means 'draw a white line from the position set by MOVE to a new position'. To determine positions, two numbers, X and Y are used as co-ordinates. The number X is a distance in units across the screen, with X=0 at the left-hand side; for the unexpanded ATOM, the maximum value of X is 64. The Y number measures distance up the screen (Y=0 at the *bottom* of the screen) and on an unexpanded ATOM this has a maximum value of 48. A fully expanded ATOM with high resolution graphics permits maximum values of X=256 and Y=192. These same co-ordinates are the two other numbers which are used for the PLOT command. Note that unless these commands are preceded by CLEAR 0 (on an unexpanded ATOM), the computer will 'lock up', and the BREAK key has to be used. The CLEAR command prepares the computer for graphics use, and must be followed by a number which specifies the resolution of the graphics.

Another addition is the use of lower-case letters (typed a, b, c etc in text, but appearing on the screen as inverted upper case, (black on white) to indicate where a GOTO or GOSUB is to go. The use of such labels speeds up transfers, but there seems little real justification for the facility.

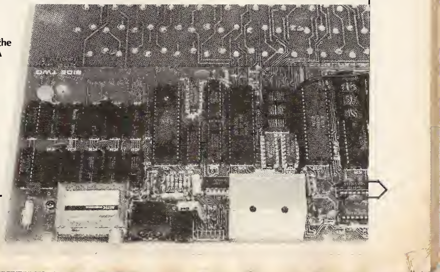
The troublesome commands for experienced programmers are the ones they will use instinctively — and therefore get wrong. The use of the single quote mark in PRINT statements is one good example of this, the use of '?' is another. Users of Microsoft BASIC in all its varieties are by now pretty well accustomed to finding that '?' has the same effect as PRINT, and is one of the few abbreviations that most computers do support. You can forget that one when you use the ATOM, because '?' is the command that replaces Microsoft BASIC's PEEK and POKE. For example, PRINT?32768 will print the value of the byte in memory address 32768; we can specify Hex address numbers by preceding the number with the hashmark (#). If we use the command : ?32768 = 65 , this is equivalent to a POKE action, placing the byte with (decimal) value 65 into the address specified by the number before the equals sign. This is, in fact, a particularly neat way of implementing PEEK and POKE, and is, if anything, an improvement on other versions. Once you can stomach the use of the words "byte indirection" for this action, you will be well away.

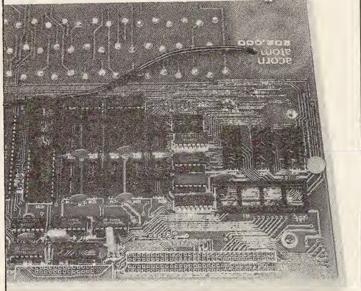
Stringing It Along

The most awkward problems are found in carrying out some procedures which, in Microsoft BASIC, are perfectly straightforward but in ATOM BASIC involve some very peculiar procedures. For some applications, including the very important educational market, this may rule out ATOM altogether. The "standard" computer for educational purposes is the RML 380Z, which uses several varieties of BASIC, all close to Microsoft standards. For my purposes, the string-handling commands of ATOM BASIC are the most off-putting — if these were reasonably normal, then the range of applications for the ATOM would be very much greater. Most of the programs which I write, either for my own pruposes or for education use, involve a lot of string handling. I am also unused to having to dimension the size of each string, neither the RML or my own TRS-80 require this, but I know that some computers do. The real problem is that the ATOM equivalents of the main string handling commands are of mind-boggling obscurity. For example, to join string B to the end of string A, we need the command: $A+LEN(A)=$B, which is not exactly so memorable as the Microsoft A$+B$. Right-string extraction, the command RIGHT$(A$,4) is, in ATOM BASIC, $B=$A+4, again not exactly memorable. LEFT$ is even worse — the LEFT$(A$,4) command in ATOM BASIC is $A+4="", and I challenge anyone to find a logical way of remembering that one! The MID$ command is simulated by combining the left and right extractions, and the example shown in the manual is:

```
10 $A = "ATOMBASIC"
20 $A+5="";$A=$A+1
```

which extracts TOMB. Certainly, once


The empty sockets behind the UHF modulator are for the high-res graphics. A total of 6K can be fitted.



Above: More empty sockets, this time it's the I/O and user memory. A total of 5K can be added to the existing 1K.

ABS (A.)	GOSUB (GOS.)	*PTR
AND (A.)	GOTO (G.)	PUT
BGET (B.)	IF	REM
BPUT (B.)	INPUT (IN)	RETURN (R.)
CH	LEN (L.)	RND (R.)
CLEAR	LET (optional)	RUN
COUNT (C.)	LINK (LI.)	SAVE (SA.)
DIM	LIST (L.)	SGET (S.)
DRAW	LOAD (LO.)	*SHUT (SH.)
DO	MOVE	SHUT (SP.)
END (E.)	NEW (N.)	STEP (S.)
EXT (E.)	NEXT (N.)	THEN (o tional)
*FIN (F.)	OLD	TO
FOR (F.)	OR	TOP (T.)
FOUT (FO.)	PLOT	UNTIL (U.)
GET (G.)	PRINT (P.)	WAIT

Table 1. ATOM BASIC commands, abbreviations in brackets. Those marked with an "*" are not available on the extended ATOM.

COLOUR	ABS	FLT (F.)
FDIM	ACS	HTN (H.)
FIF	ASN	LOG (L.)
FINPIT (FIN.)	ATN	PI
FPRINT (FP.)	COS (C.)	RAD
FPUT	DEG (D.)	SGN
FUNTIL (FU.)	EXP (E.)	SIN
STR	FGET	SQR
TAN (T.)	VAL (V.)	

Table 2. Extra commands available with the floating point ROM.

you get used to it, it's not as bad as it appears at first sight, but what on earth was wrong with LEFT$, RIGHT$ and MID$?

The statements READ and DATA which are usually among the first BASIC commands that a beginner learns in a computing course simply don't exist on the ATOM, and the methods which can be used to achieve the same effects are so complex that it's better to forget about this type of command altogether.

I must conclude, sadly, that if your interests are in string handling, the ATOM is not really a suitable computer to learn on, and you certainly could not transfer software from the ATOM to a 'conventional' computer. If, on the other hand, your interests are in graphics and games (and I suppose there are still some people interested in games) then the features of the ATOM may appeal very strongly. You have, after all, in the PLOT, MOVE and DRAW instructions, a set of graphics capabilities which would cost you a fortune on other machines (try asking the price of a 380-Z, for example) and, with a relatively simple low-cost expansion you can do high-resolution work.

Before we leave the subject of BASIC programming it's worth looking at the way in which ATOM stores its BASIC instructions. The unexpanded ATOM stores its programs starting at address 33282 (decimal), and the form of the line is straightforward. The first byte or pair of bytes represent the line number, using the most significant byte first (omitted if zero). For example, line 10 appears as 10, but line 300 is stored as 01 44 (1 x 256 + 44 = 300). The instructions in the line appear in their ASCII code form, with a carriage return (ASCII 13) and a zero to mark the end of the line. The last line (END) is terminated by a carriage return (13), and the last two bytes are 255 164 end markers — these appear even if the END has been omitted.

The simple construction makes it very easy to synthesise or modify lines from a running program by POKEing values directly into memory (OLD then has to be used to ensure that the pointers are correctly set), but on the other hand, it is wasteful of memory. Microsoft BASIC, as used on TRS-80 and others, stores each BASIC command as a single byte. By not doing this, the ATOM can include a set of half-tone (grey) graphics, but on a machine which has such a small amount of memory (½ K unexpanded) in standard form, the ROM storage which would have been needed to produce the 'tokens' would surely have been worthwhile. When I first read the advertisements for a certain other computer announcing that it used single-byte command words, I wondered who they could possibly be getting at — now I know! In mitigation, however, it must be said that the ATOM method allows a printer to be interfaced without re-designing the ROM, and memory can be saved to a considerable extent, as on the old TRS-80 Level 1, by using the abbreviations.

Assembled Conclusions

Finally, a very potent reason for buying an ATOM is the least-expected one. The ROM includes a 6502 assembler, which permits assembly language instructions in mnemonic form to be typed in and assembled into machine code. I spend some of my time teaching 6502 programming using a KIM-1 so that the students have to assemble 'manually', and punch code in Hex. For very little extra cost, the ATOM would allow users to type in assembler language directly, with the added advantage of a very reliable cassette interface which really does work with low-price recorders. The provision of the assembler gives such an overwhelming advantage to the ATOM that I can see this as being a major market 'plus' for the machine — it is certainly one facility which will figure very largely in my own use.

To sum up, then, the ATOM is a strange mixture of a machine, designed with very few attempts for it to be compatible with other machines. Some of its features may make it almost irresistible to you — the excellent graphics, and the assembler are two particularly strong points. Other features, in particular the string handling, may make it rather unattractive. On the whole, it is never a boring machine to use, and the more I use it the better I like it. I greatly look forward to expanding the memory and adding some of the many standard ACORN interfaces.